VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Tvorba SW podpory testování v SWI
Testing Tool for SWENG Course

2011 Martin Cendelín

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

15. 8. 2011 Martin Cendelín

Abstrakt: Bakalářská práce se zabývá tématem testováním studentů v předmětu Úvod do softwarového inženýrství, se zaměřením na oblast jazyka UML. Práce obsahuje krátký popis předmětu, pro který je aplikace určena, stručný přehled programů, používaných k prověření znalostí studentů a také popis technologií použitých při tvorbě aplikace. Dále je součástí práce analýza požadavků, popis nejdůležitějších principů, použitých při tvorbě aplikace a jednoduchý popis chování aplikace.

Abstract: The bachelor thesis describes theme of testing students in course Introduction to software engineering, with focus on UML language. The work contains a short description of the course for which the application is determined, a brief overview of the programs used to test students' knowledge and description of technologies used to create applications. Next part of the work is requirements analysis, a description of the main principles used to create applications and simple description of application behavior.

Klíčová slova: **Sofwarové inženýrství, UML, Silverlight , Tvorba testů**

Key-words: **Software engineering, UML, Silverlight, Creation tests**

Poděkování:
Děkuji svému vedoucímu, Ing. Svatopluku Štolfovi, za osobní přístup, odborné vedení mé práce, cenné rady, pomoc a čas, který mi věnoval.

Seznam použitých zkratek a symbolů

UML - Unified Modeling Language

XAML - Extensible Application Markup Language

XML - Extensible Markup Language

XHML - Extensible Hypertext Markup Language

CB – Code Behind

Obsah

1 Úvod

Cílem této bakalářské práce je vytvoření webové aplikace schopné otestovat znalosti získané během kurzu Úvod do softwarového inženýrství.

Na začátek se zmíním o kurzu Úvod do softwarového inženýrství a nastíním požadavky na výslednou aplikaci.

Před samotným návrhem implementace bylo potřeba udělat výzkum v oblasti programů pro testování studentů. V této kapitole uvedu přehled pěti nejznámějších programů a uvedu jejich výhody a nevýhody.

V následující kapitole se zmíním o použitých technologiích. Nejprve popíšu průběh volby technologie, ve které jsem se rozhodl výslednou aplikaci implementovat. Dále popíšu základy této technologie. Nakonec se ve zkratce zmíním o technologii použité pro datovou vrstvu a vysvětlím základní prvky jazyka UML, které budou součástí testů ve výsledné aplikaci.

Po popisu teoretické části práce se dostanu k samotné aplikaci. Na začátku kapitoly provedu analýzu požadavků, poté popíšu několik důležitých principů, které jsem použil při implementaci a ve zkratce popíšu chování aplikace.

2 Úvod do softwarového inženýrství

2.1 Cíle kurzu

Cílem předmětu Úvod do Softwarového inženýrství je uvést studenty do disciplíny zabývající se problematikou vývoje rozsáhlých softwarových systémů. Tento předmět je úvodem do problematiky tvorby software z hlediska inženýrských metod. Objektově orientovaný přístup a jazyk UML je použit jako základ prezentovaných metod.[1].

Obsah předmětu Úvod do softwarového inženýrství: [2]

1. Základní pojmy
   1.1. Definice softwarového inženýrství
   1.2. Schéma procesu vývoje softwarového díla
2. Softwarový proces
   2.1. Definice softwarového procesu
   2.2. Základní typy softwarového procesu
   2.3. Process RUP, jeho cykly, fáze a iterace
   2.4. Základní a podpůrné toky činností
   2.5. Jazyk UML
3. Byznys modelování
   3.1. Diagramy aktivit a tříd jazyka UML
   3.2. Metoda BPM
4. Specifikace požadavků
   4.1. Aktéři a funkční specifikace pomocí případů použití
   4.2. Diagramy případů užití
   4.3. Definice pojmu objekt
   4.4. Vztahy mezi objekty a jejich interakce
5. Analýza a návrh
   5.1. Modely a jejich diagramy
   5.2. Definice pojmu třída
   5.3. Vztahy mezi třídami a objekty
   5.4. Diagramy tříd jazyka UML
   5.5. Specifikace dynamického chování
   5.6. Strukturování softwarového systému
   5.7. Návrh a jeho cíle
   5.8. Architektura výsledného systému
   5.9. Návrhové vzory a aplikační rámce
   5.10. Model nasazení

6. Implementace
   6.1. Mapování elementů logického modelu na komponenty
   6.2. Zdrojové, binární a spustitelné komponenty
   6.3. Diagramy nasazení
7. Testování a nasazení softwarového produktu
   7.1. Cíle procesu verifikace a validace
   7.2. Modely testování
   7.3. Nasazení softwarového systému

Softwarové inženýrství je inženýrská disciplína zabývající se praktickými problémy vývoje rozsáhlých softwarových systémů.[2]

2.2 Zkouška z kurzu

Zkouška je písemná, je v ní několik otázek, každá rozdělena na dvě části. První část otázky tvoří jeden ze čtyř druhů diagramu jazyka UML (diagram aktivit, sekvenční diagram, diagram případu užití, nebo diagram tříd). Student musí v této části využít nabytých znalosti z oblasti jazyka UML a správně doplnit chybějící části diagramu. Druhá část otázky je tvořena klasickou otázkou s několika možnými odpověďmi, týkající se libovolné části učiva. Body z první části otázky jsou násobeny dvěma, pokud student správně odpoví na druhou část otázky.

2.3 Shrnutí

Tato kapitola stručně popisuje kurz Úvod do softwarového inženýrství a průběh zkoušení a zároveň ukazuje kostru toho, jak by měla výsledná aplikace pracovat.

3 Přehled programů pro testování studentů

3.1 Úvodní poznámky

Cílem této části je podat zprávu o dostupných aplikacích pro tvorbu testů. Na úvod je třeba říct, že filosofie tvorby testů je u všech níže uvedených programů téměř stejná. Hlavními kameny jsou tyto čtyři pojmy:

- **Banka otázek** – do banky otázek vkládáme všechny vytvořené otázky. Pro přehlednost je zakládáme do kategorií
- **Test** – do testů vkládáme otázky z banky otázek. Pokud máme otázky rozděleny do kategorií, můžeme vkládat celé kategorie a tím si práci výrazně zjednodušíme.
- **Uživatelé** – zde vkládáme různé uživatele, od studentů, které budeme testovat, až po učitele kteří budou mít práva k vytváření a modifikování testů.
- **Studijní skupiny (nebo také kurzy a podobně)** – zde vkládáme jednotlivé studenty a zároveň ke každé skupině přiřazujeme určité testy, čímž určujeme kdo má jakým testem projít.

3.2 Moodle

- Modular Object-Oriented Dynamic Learning Environment (Modulární objektově orientované dynamické prostředí pro výuku)
- Licence-freeware

3.2.1 Instalace a spuštění

- Požadavky: PHP 4.3.0, MySQL 4.1.16 or Postgres 8.0 or MSSQL 9.0 or Oracle 9.0
- Instalace neprobíhá pomocí klasického instalátoru, ale tak že po stažení zabaleného souboru jej rozbalíme a celou složku moodle nahrajeme na náš server. Při první návštěvě moodlu je potřeba ještě dokončit instalaci, kterou nás provede přehledný

průvodce. Po dokončení instalace je moodle připraven k použití. Program je kompletně v češtině.

3.2.2 Práce v programu

Moodle je kompletní elearningový systém a vytváření testů je pouze jeho součástí. Na tento fakt je třeba myslet a v systému se nejprve zorientovat. Přes vytvoření kurzu a režim úprav se dostaneme až k samotnému testu. Do vytvořeného testu poté vkládáme pomocí banky úloh jednotlivé otázky. Možnosti nastavení testů a jednotlivých otázek je velké množství, vše je však velmi přehledně uspořádáno, takže i nezkušený uživatel se dokáže rychle zorientovat.

3.2.3 Zhodnocení

Moodle vyniká hlavně obrovským množstvím možností, které jsou doprovázeny celkem přehledným zpracováním. Další výhodou je práce rovnou v prohlížeči, díky čemuž nemusíme instalovat žádného klienta. A v neposlední řadě je obrovskou výhodou, že je celý systém freeware

3.3 Test Generator

- Licence - 30-ti denní trial verze, poté je nutné zakoupit licenci

3.3.1 Instalace a spuštění

Instalace probíhá pomocí klasického instalátoru, kdy je po několika obvyklých dialogových oknech spuštěna instalace. Po instalaci je nám k dispozici několik spustitelných souborů, mezi něž patří hlavně Tester (rozhraní pro studenty, kde absolvují testy) a TestGenerator (rozhraní pro vytváření testů). Program nepodporuje češtinu.

3.3.2 Práce v programu

Úvodní okno programu je na první pohled velice přehledné a hned je jasné k čemu který panel slouží. Co je, však méně přehledné jsou velmi malé ikony, u kterých není příliš jasné, co vyjadřují. Celkem máme k dispozici 11 druhů otázek.

3.3.3 Zhodnocení

Test Generator vyniká hlavně přehledností uživatelského rozhraní.

3.4 Test Maker

- Licence - 30-ti denní trial verze, poté je nutné zakoupit licenci

3.4.1 Instalace a spuštění

Instalace probíhá pomocí klasického instalátoru, kdy je po několika obvyklých dialogových oknech spuštěna instalace. Po instalaci jsou nám k dispozici dva spustitelné soubory: Tester (rozhraní pro studenty, kde absolvují testy) a TestMaker (rozhraní pro vytváření testů). Program nepodporuje češtinu.

3.4.2 Práce v programu

Program pracuje ve dvou základních režimech:

- Test editor - slouží k samotnému vytváření testů a testových otázek
- Test admin - slouží ke správě studentů, studijních skupin a k jejich přiřazování k vytvořeným testům

Je možné použít pouze 4 druhy otázek-Jedna správná odpověď, více správných odpovědí, odpověď ve formě eseje a správné seřazení odpovědí.

3.4.3 Zhodnocení

Program je podobně jako Test Generator dosti jednoduchý a přehledný. Nevýhodou ale je malý počet druhů otázek a nutnost přepínat mezi jednotlivými mody programu.

3.5 Quedoc Quiz Maker

- Licence - freeware

3.5.1 Instalace a spuštění

Instalace probíhá pomocí klasického instalátoru, kdy je po několika obvyklých dialogových oknech spuštěna instalace. Po instalaci jsou nám k dispozici dva spustitelné soubory: Qdoc Quiz Player (rozhraní pro studenty, kde absolvují testy) a Qdoc Quiz Maker (rozhraní pro vytváření testů). Program nepodporuje češtinu.

3.5.2 Práce v programu

Hned po instalaci jsme vyzváni ke zvolení obtížnosti – ne k obtížnosti testu ale k jakési „složitosti“ uživatelského rozhraní. Čím větší obtížnost tím víc možností uživatel má, ovšem s rostoucí obtížností stoupá i chaotičnost celého rozhraní, která je oproti ostatním programům zde uvedeným velmi velká a uživatel se těžce orientuje.

3.5.3 Zhodnocení

Program vyniká hlavně velkými možnostmi, které zde jsou na úkor přehlednosti. Další zajímavostí je vizuálně velmi zajímavé prostředí.

3.6 DoTest 4

- Český zástupce testovacích programů
- Licence - 30-ti denní trial verze, poté je nutné zakoupit licenci

3.6.1 Instalace a spuštění

Instalace probíhá pomocí klasického instalátoru, kdy je po několika obvyklých dialogových oknech spuštěna instalace. Po dokončení instalace máme k dispozici jeden spustitelný soubor. Podle způsobu přihlášení získáme různá práva (administrátor/student). Program je kompletně v češtině.

3.6.2 Práce v programu

Po prvním spuštění je uživatel vyzván k volbě jednoho z pěti módů:

- tvorba databází otázek
- sestavování testů
- tisk variací testů
- zkoušení
- výsledky zkoušení

Práce v programu je velmi přehledná, avšak na výběr máme pouze 5 druhů otázek (klasické, obrázkové, přiřazovací, uspořádavací a doplňovací)

3.6.3 Zhodnocení

Program vyniká hlavně svou přehledností. Mezi nevýhody patří malé množství druhů otázek a nutnost přepínání mezi jednotlivými módy.

3.7 Shrnutí

V této kapitole jsme si stručně popsali několik nejběžnějších programů na testování studentů. Všechny tyto program jsou si dost podobné. Liší se hlavně možnostmi testů a testovacích otázek a také přehledností. Z těchto hledisek mě nejvíce zaujal program Moodle, který představuje velmi komplexní možnost testování studentů s obrovskými možnostmi a přesto zachovanou přehledností při vytváření testů. Dalším důležitým poznatkem této části bylo získání představy o základní filosofii tvorby testů, kterou jsem mohl využít při další implementaci.

4 Použité technologií

První otázkou před započetím práce byla volba technologií pro implementaci programu. Hned na první pohled je jasné, že tento typ aplikace je nejlepší vytvořit jako webovou aplikaci.

4.1 Webová aplikace

Webová aplikace je taková aplikace, která je poskytovaná uživatelům z webového serveru přes internet.

Internet v dnešní době snad ani není třeba představovat. Je celosvětová síť navzájem propojených počítačových sítí. Počítače mezi sebou komunikují a sdílejí data pomocí rodiny protokolů TCP/IP. V dnešní době je internet standardem, stejně jako základní znalosti ovládáni webu, obzvláště pak v universitních podmínkách.

V následujícím odstavci popíšu několik výhod, kvůli kterým jsem si vybral právě webovou aplikaci. Prvním důležitou výhodou oproti klasické aplikaci je způsob distribuce. Aplikace je umístěna na webu a tudíž není nutné ji instalovat na počítače, na kterých bude zkouška probíhat. Také učitelé budou moct tvořit diagram na různých počítačích bez ohledu na instalovaný software. S tímto částečně souvisí i otázka aktualizace systému. Díky tomu, že je aplikace umístěna na webu, se aktualizace projeví všem uživatelům současně. Není tedy třeba instalovat nové verze programu na klientský počítač. Dalším faktorem je nezávislost na operačním systému. Webové stránky se zobrazují na všech operačních systémech téměř stejně. Zobrazení však může ovlivnit webový prohlížeč. Jedinou větší nevýhodou webových aplikací je náročná instalace. Webová aplikace musí být nainstalována na webovém serveru, který podporuje všechny potřebné webové technologie, jako například PHP, ASP.NET a další.

4.2 .NET Framework

Dalším krokem bylo zvolit vhodnou technologii pro implementaci aplikace. Během studia velmi zaujala rodina technologií .NET, které taktéž podporují vývoj webových aplikací.

.NET Framework je platforma pro vytváření aplikací od společnosti Microsoft. Tato platforma umožňuje jednoduše vyvíjet vizuálně ohromující aplikace, s bezpečnou komunikaci a možnost modelovat škálu podnikových procesů. [3]

.NET Framework se skládá z: [3]

- Common Language Runtime - poskytuje abstraktní vrstvu nad operačním systémem
- Base Class Libraries - předem připravený kód pro běžné programování úloh na nižší úrovni
- Development frameworks and technologies - opakovaně použitelné a přizpůsobitelné řešení pro složitější úkoly

Pro vývoj jsem se rozhodoval mezi dvěma webovými technologiemi, se kterými jsem během studia setkal: ASP.NET a Silverlight. Nakonec jsem zvolil technologii Silverlight. V následující kapitole tuto technologii popíšu a uvedu několik důvodů, proč jsem si ji zvolil.

4.3 Silverlight

4.3.1 Úvodní informace

Microsoft® Silverlight™ je nejmodernější technologie pro internetové prohlížeče. Je to platforma určená pro tvorbu dynamického online obsahu a interaktivní práce s ním. Kombinuje text, vektorovou i bitmapovou grafiku, animace a video.[4]

Jde o multiplatformní implementaci .NET Frameworku určenou pro tvorbu RIA aplikací. RIA (rich internet application) je webová aplikace, běžící v prohlížeči, která má některé vlastnosti desktopových aplikací. Další RIA aplikací je například Adobe Flash.

Silverlight je poměrně mladá aplikace. První verze byla uvolněna v roce 2007. Již od začátku bylo hlavním cílem přenést funkcionalitu Windows do webového prohlížeče a tak použít nových technologií při vytváření webových prezentací[5]. Důležitým charakteristickým rysem je oddělení grafického návrhu od logiky. Grafický návrh je tvořen v jazyce XAML, který je založen na technologii XML, a později se o něm ještě zmíním. Logika může být implementována v jazyce C# nebo Visual Basic.

Samotný Silverlight se instaluje pomocí malé stažitelné komponenty (plug-in) umožní interaktivní ovládání her nebo aplikací a přehrávání multimédií ve většině současných webových prohlížečů (Internet Explorer, Firefox, Safari, Opera, Chrome) na platformách Windows a Mac OS X. Na Linuxu je dostupný pod názvem Moonlight, vyvinutý společností Novell.[4]

4.3.2 Silverlight vs. ASP.NET

Obě technologie toho mají spoustu společného. U obou je grafický návrh oddělený od logiky. Logika může být u obou technologií implementována v jazyce C# nebo Visual Basic. V grafickém návrhu se už obě technologie liší. U ASP.NET je grafická část implementována v jazyce XHTML, zatímco u Silverlightu jde o jazyk XAML.

Daleko významnějším rozdílem je způsob komunikace se serverem. Zatímco ASP.NET posílá na server požadavky na jednotlivé stránky a server podle požadavků uživatele vrací stránky jako celek, u Silverlightu jde o celistvou aplikaci na straně uživatele, která se serverem komunikuje jen kvůli dílčím požadavkům.

Dalším rozdílem a hlavním důvodem proč jsem dal přednost Silverlightu před ASP.NET je efektivita grafického návrhu. V ASP.NET je s jednotlivými prvky stránky nakládáno jako v klasickém XHTML, což je pro tvorbu UML diagramů velice nepraktické. Dodatečně jsem po krátkém hledání zjistil, že i v ASP.NET se dají podobné aplikace vytvořit. Nicméně jako mnohem efektivnější se mi zdál grafický návrh Silverlightu. Ten může s prvky nakládat mnoha způsoby podle zvoleného layoutu. O layoutech se ještě více zmíním.

Nakonec je třeba zmínit, že každá Silverlight aplikace v sobě obsahuje dva projekty. Jeden je samotný Silverlight a druhý je ASP.NET aplikace. Výsledkem těchto dvou projektů je v podstatě Silverlight komponenta v ASP.NET stránce. Oba projekty spolu můžou do jisté míry komunikovat, ale dle mého názoru je tato komunikace dost komplikovaná a doufám, že v nových verzích dojde ke zjednodušení.

4.3.3 XAML

XAML je značkovací jazyk, syntakticky vycházející z jazyka XML. Hlavním znakem tohoto jazyka jsou tagy. Tyto tagy jsou zapisovány ostrými závorkami a tím vytvářejí elementy. Elementy můžou být buď párové (uvnitř elementu se pak může nacházet nějaké parametry, nebo další elementy), nebo nepárové.

Příklad nepárového elementu:

```
<Ellipse Fill="Black"/>
```

Příklad párového elementu:

```
<Ellipse>
      <Ellipse.Fill>
            <SolidColorBrush Color="Blue"/>
      </Ellipse.Fill>
</Ellipse>
```

Z příkladů je vidět, že popis takovéhoto elementů je velmi přehledný a jednoduchý. Při větším množství elementů a atributů se přehlednost ztrácí a člověk přestává být schopen efektivně číst v kódu. Nicméně číst v takto složitém kódu už není nutné, jelikož většina vývojových prostředí má design editor, díky kterému vývojář vidí, jak budou elementy vypadat na výstupu.

V každé Silverlight aplikaci se nachází dva XAML soubory. Jeden z názvem app.xaml a druhý s názvem MainPage.xaml.

App.xaml je, krom jiného, určen k definici šablon, stylů a celkově všech nastavení aplikace. Dále do něj můžeme zařadit předdefinovaných animace. MainPage.xaml je určen k vytváření samotného obsahu aplikace.

4.3.4 Spojení logické a designové části

Každá XAML stránka má k sobě přiřazený stejně pojmenovaný soubor s logikou (koncovka je závislá na použitém jazyku). Tento druhý soubor, označovaný jako „Code Behind“ vytváří logiku dané stránky.

Všechny elementy v XAML souboru jsou zároveň i třídami v CB. Elementy je tedy možné vytvářet i měnit dynamicky v této části kódu.

XAML:

```
<Ellipse Name="Elipsa"/>
```

Code behind(C#)

```
Elipsa.Fill = new SolidColorBrush(Colors.Black);
```

Dalším způsobem komunikace mezi design částí a CB spočívá v událostech, které má každý element připravené. Těchto událostí je poměrně dost, od pohybu myší, přes stisknutí klávesy až po ztrátu focusu. U každého elementu se můžou události měnit. Přiřazením metody k handleru určitého objektu docílíme toho, že se metoda spustí ve chvíli, kdy dojde k události.

```
<Button x:Name="Tlacitko" Click="Tlacitko_Click"/>
```

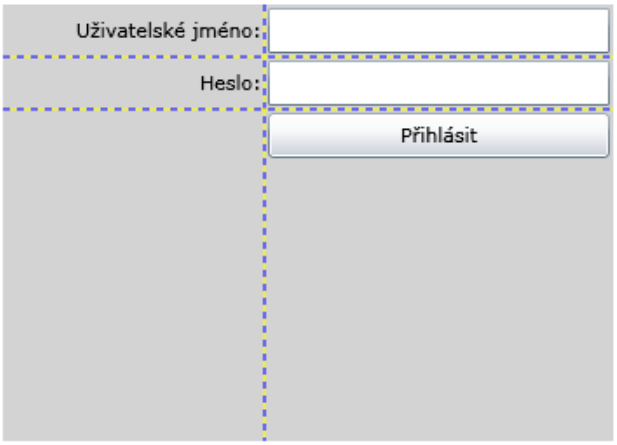
4.3.5 Základní elementy

Jazyk Silverlight podporuje velmi velké množství elementů, mezi něž patří různé geometrické tvary, textová pole, tlačítka, datové kontejnery, a mnoho dalších. Důležitou skupinou elementů, o které bych se rád zmínil, jsou takzvané „layouty“.

Tyto layouty určují způsob řazení elementů na obrazovce. Do každého layoutu může být vnořen další layout. Nyní uvedu přehled nejběžnějších layoutů a ke každému uvedu stručný popis.

4.3.5.1 Grid

Grid je výchozím typem layoutu při tvorbě silverlight aplikace. Je to v podstatě jakýsi druh tabulky. Nejdříve je třeba nadefinovat sloupce a řádky a elementy se poté umisťují do těchto políček. Je vhodný pro základní rozvržení stránky.



Obr. 1: Ukázka Gridu, převzato z [6]

4.3.5.2 StackPanel

Ve StackPanel se elementy řadí za sebou a to horizontálně, nebo vertikálně. Rozměry jednotlivých řádků, nebo sloupců jsou dány rozměrem elementu, který je do StackPanelu vložen. Je vhodný v případě, kdy chceme dynamicky přidávat elementy. Na obrázku 2 vidíme prvky skládající se pod sebe s různým zarovnáním (vlevo, na střed, vpravo).

Obr. 2: Ukázka StackPanelu, převzato z [6]

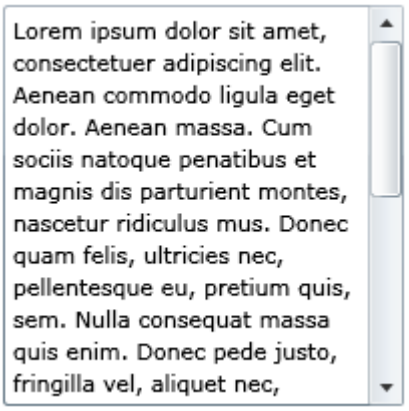
4.3.5.3 Canvas

Canvas je nejsvobodnější ze všech layoutu. Pozice každého prvku je definována jeho reálnou pozicí (v pixelech) v canvasu. K přiřazené pozice slouží elementům parametry Top, Left a ZIndex. Top je vzdálenost od horního okraje canvasu, podobně jako Left určuje vzdálenost od levého okraje. ZIndex je ekvivalentem zetové souřadnice v kartézské soustavě souřadnic. Element, který má nižší hodnotu tohoto parametru, je překryt prvkem s vyšší hodnotou. Díky této volnosti v pozicích prvků je Canvas ideální pro implementaci tvorby diagramu.

Obr. 3: Ukázka Canvasu, převzato z [6]

4.3.5.4 ScrollView

ScrollView je velmi odlišný od ostatních typů layoutů. Jeho úkolem je vytvořit rolovací lištu, tam kde je jasné, že se celý obsah nevejde do rozměru, který bychom potřebovali. Do tohoto layoutu je možné vložit pouze jeden element (nejlépe nějaký jiný layout, do kterého poté vložíme ostatní elementy)



Obr. 4: Ukázka ScrollView

4.3.6 User contol

Jednou ze silných stránek technologie Silverlight je možnost tvorby vlastních elementů. Takový element se nazývá UserControl. Každý UserControl vypadá jako klasická MainPage stránka, a je možné v něm dělat to samé jako v této stránce. Po uložení se takovýto UserControl zobrazí v panelu nástrojů a je možné s ním pracovat jako s jakýmkoliv jiným elementem. Této vlastnosti jsem hodně využil při implementaci diagramů a více se o tomto tématu zmíním níže.

4.3.7 Vývojové prostředí

4.3.7.1 Microsoft Expression Blend 4

Toto vývojové prostředí je vytvořeno speciálně pro tvorbu Silverlight aplikací. Jeho síla spočívá hlavně v grafickém návrhu aplikace. Oproti Visual Studiu poskytuje různé nástroje a usnadnění pro editaci XAML části kódu, mezi něž patří design editor, nebo také nástroj pro tvorbu animací. Je možné implementovat i logickou část, avšak Visual Studio v tomhle ohledu poskytuje mnohem efektivnější možnosti práce. Většinu XAML kódu jsem implementoval právě v tomto prostředí.

4.3.7.2 Microsoft Visual Studio 2010

Jde o komplexní nástroj pro programování ve všech jazycích z rodiny .NET. Podobně jako v Blendu je i zde grafický editor pro tvorbu XAML části kódu, ovšem editor v Blendu je podlé mé zkušenosti přehlednější a efektivnější. Jak už jsem se výše zmínil, Visual Studio vyniká spíše v implementaci logické části. Z toho důvodu jsem také většinu logiky programoval ve Visual Studiu.

4.4 SQL

Pro datovou vrstvu systému jsem se rozhodl využít relační databázi založenou na jazyce SQL, konkrétně Microsoft SQL Server 2008 R2. Pro práci v databázi jsem využil nástroj SQL Server Management Studio. SQL kódy a návrh databáze jsou umístěny v přílohách na CD.

4.5 UML

UML je jazyk umožňující specifikaci, vizualizaci, konstrukci a dokumentaci artefaktů softwarového systému.[2]

Pod jednotlivými charakteristikami jazyka UML rozumíme:[2]

- Specifikace vyjadřuje zásadu vytvoření přesných, jednoznačných a úplných modelů softwarového procesu.
- Vizualizace znamená, že se jedná o grafický jazyk.
- Konstrukce odpovídá požadavku přímého napojení jazyka na širokou škálu programovacích jazyků.

Jazyk UML používáme před samotnou implementací aplikace. Slouží nám k zaznamenávání myšlenek a návrhů, které kreslíme do diagramů.

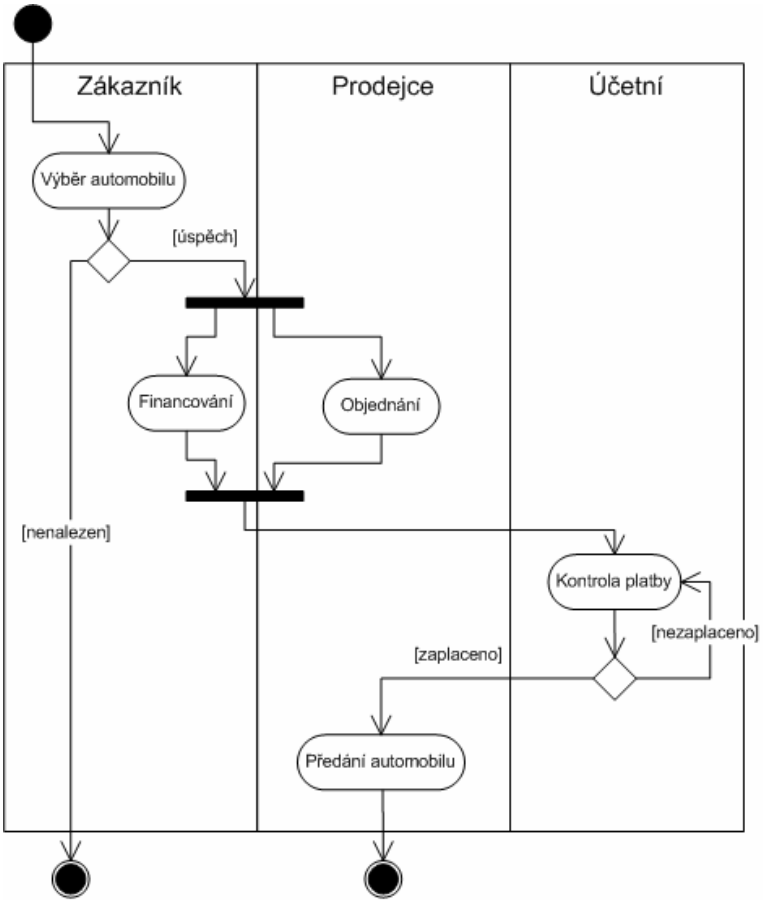
K vytváření jednotlivých modelů systému jazyk UML poskytuje celou řadu diagramů umožňujících postihnout různé aspekty systému. Jedná se celkem o čtyři základní náhledy a k nim přiřazené diagramy:[2]

1. *Funkční náhled*
   a. Diagram případů užití
2. *Logický náhled*
   a. Diagram tříd
   b. Objektový diagram
3. *Dynamický náhled popisující chování*
   a. Stavový diagram
   b. Diagram aktivit
   c. Interakční diagramy
      i. Sekvenční diagramy
      ii. Diagramy spolupráce
4. *Implementační náhled*
   a. Diagram komponent
   b. Diagram rozmístění

V následujících podkapitolách se budu věnovat čtyřech základním diagramům, které se v předmětu Úvod do softwarového inženýrství používají k testování vědomostí studentů.

4.5.1 Diagram aktivit

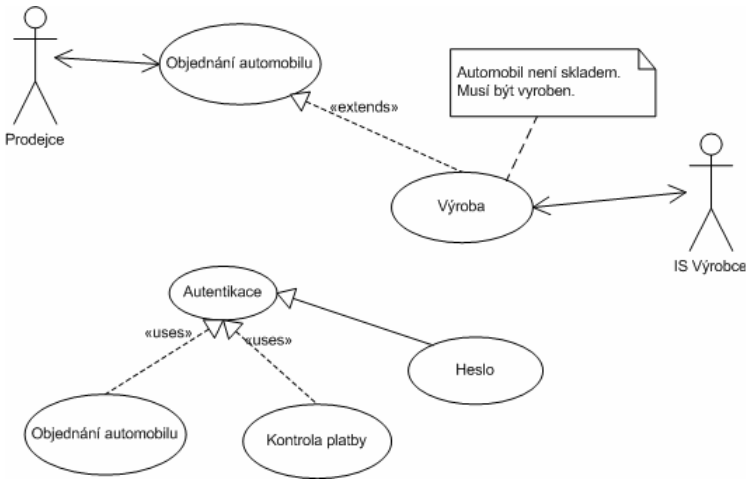
Diagram aktivit popisuje jednotlivé procesy pomocí aktivit reprezentujících jeho (akční) stavy a přechody mezi nimi. [2]



Obr. 5: Ukázka diagramu aktivit, převzato z [2]

4.5.2 Diagram případu užití

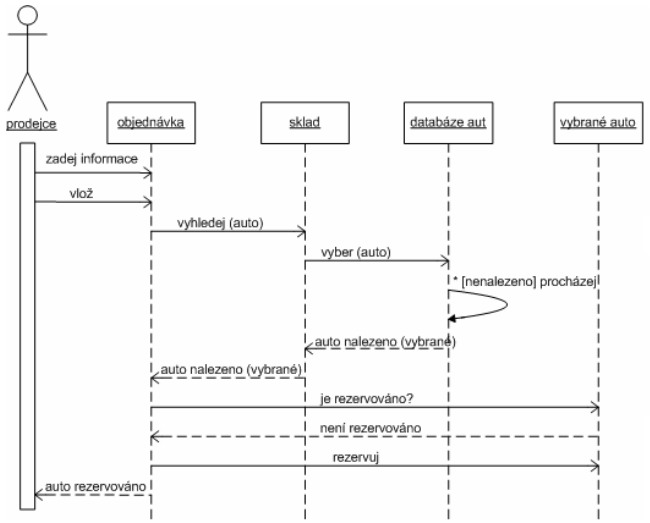
Účelem diagramu případů užití je definovat co existuje vně vyvíjeného systému (aktéři) a co má být systémem prováděno (případy užití). Notace používaná diagramy případů užití je tvořena grafickými symboly reprezentující aktéry a případy užití v jejich vzájemných vazbách (obr. 6).[2]



Obr 6. Ukázka Diagramu případu užití, převzato z [2]

4.5.3 Sekvenční diagram

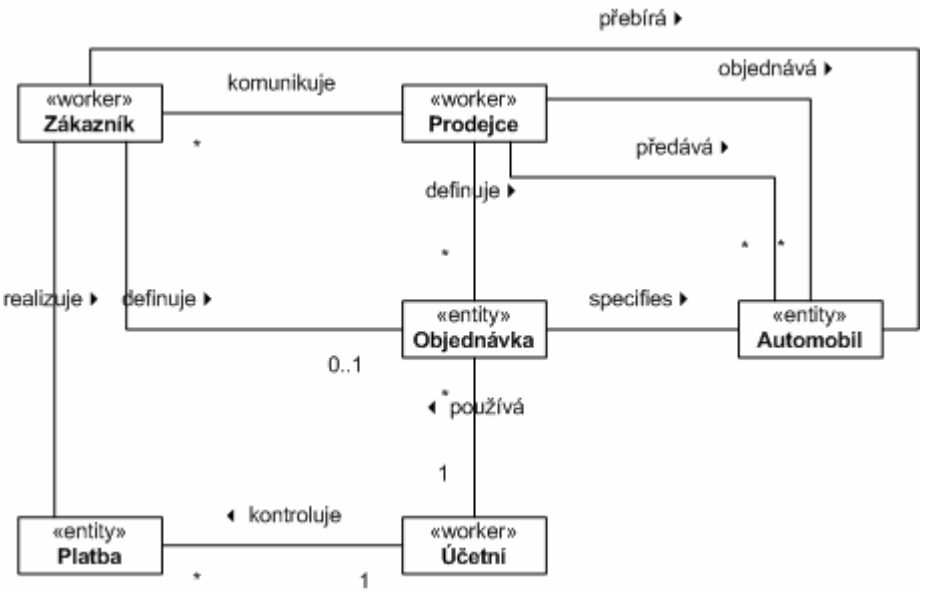
Slouží k zachycení vzájemné interakce z časového hlediska.



Obr. 7: Ukázka sekvenčního diagramu, převzato z [2]

4.5.4 Třídní diagram

Slouží k zachycení statické struktury. Je tvořen elementy, které jsou mezi sebou navzájem propojeny vazbami, které vyjadřují fyzické, nebo konceptuální spojení těchto elementů.



Obr. 8: Ukázka diagramu tříd, převzato z [2]

4.6 Shrnutí

Tato kapitola pojednává o základech nejdůležitějších technologií, které byly v rámci aplikace použity. Speciální pozornost jsem věnoval Silverlightu, jelikož většina aplikace bude napsána implementována právě touhle technologií.

5 Aplikace UML Tester

5.1 Základní informace

Aplikace UML Tester je webová aplikace, založená na technologii Silverlight. Při její tvorbě byly použity nástroje Microsoft Visual Studio 2010 a Microsoft Expression Blend 4. Datová vrstva je založena na relační databázi s využitím jazyka SQL.

5.2 Analýza požadavků

5.2.1 Záměr

Program tvořený v rámci této práce má za úkol ulehčit tvorbu testů ke zkoušce z předmětu Úvod do softwarového inženýrství, která doteď probíhá formou písemných testů. Dalším důvodem zavedením tohoto systému je okamžité vyhodnocování výsledků testů.

5.2.2 Rozsah

Systém by měl umožňovat tvorbu a editaci testových otázek a testů a jejich následné testování studentů, pomocí těch testů a následné archivování výsledků. Dále by neměla chybět možnost prohlédnout si studenty vyplněné testy.

5.2.3 Popis doménové oblasti

Každý učitel bude moct vytvářet a spravovat testy a testové otázky.

Každý test se skládá z několika testových otázek.

Každý otázka se skládá z dvou podotázek. První podotázka se týká doplňování chybějících částí v jednom ze čtyř druhů UML diagramu (diagram aktivit, diagram případu užití, sekvenční diagram a diagram tříd). U první podotázky učitel nastaví, kolik bodů, za jakou chybu bude odečteno. Druhá podotázka bude ve formě slovní úlohy s několika možnými

odpověďmi. Správná odpověď je vždy jedna. Body z první podotázky budou násobeny dvěma, pokud student zodpoví správně i druhou otázku.

5.2.4 Aktéři

- Učitelé – V systému mohou vyvářet a editovat testy a testové otázky, přiřazovat studenty k jednotlivým testům a prohlížet vykonané testy.
- Studenti – V systému mohou vyplňovat přidělené testy a prohlížet si výsledky jimi dříve vykonaných testů.

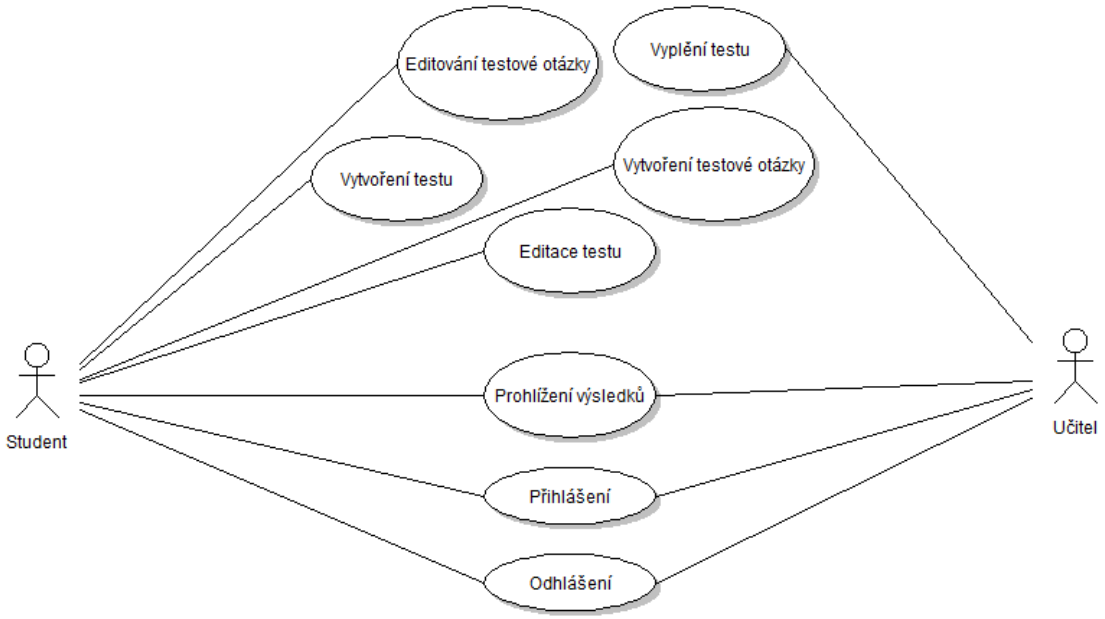
5.2.5 Funkční požadavky

1. Systém musí umožňovat přihlášení a ověření práv aktérů.
2. Systém bude umět vytvářet a editovat testové otázky.
3. Systém bude umět vytvářet a editovat testy.
4. Systém bude umět vkládat testové otázky do jednotlivých testů.
5. Systém bude umět ukládat a načítat testy a testové otázky.
6. Systém bude umět přiřazovat studentům testy, které musí vyplnit.
7. Systém bude umět testovat studenty prostřednictvím předem vytvořených testů
8. Systém bude umět vyhodnotit a uložit vyplněný test.
9. Systém bude vést evidenci dříve vyplněných testů.

5.2.6 Nefunkční požadavky

1. Ovládání systému by mělo být jednoduché a intuitivní.
2. Systém musí být spolehlivý.

5.2.7 Diagram případu užití a sekvenční diagramy



Obr. 8: Diagram případů užití

Sekvenční diagramy jsou k nalezení v příloze A.

5.3 Principy programu

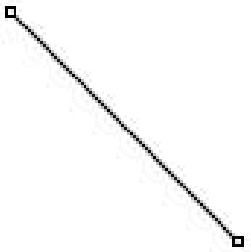
Na webu je k nalezení několik Silverlight knihoven s UserControly, které podporují práci s jazykem UML. Mezi nejrozšířenější patří například yWorks, nebo GoXam. Používání drtivé většiny těchto knihoven je ovšem placené (ceny se pohybují poměrně vysoko) a dají se použít jen v třiceti denní trial verzi. Ty, které jsou zdarma většinou nejsou kompletní a nesplňují požadavky téhle práce. Proto bylo nutné si veškeré grafické rozhranní a jeho chování naimplementovat. V následující podkapitole uvedu několik důležitých principů, které jsem během implementace musel vymyslet a použít.

5.3.1 Elementy a vztahy mezi nimi

Úplně prvním problémem při tvorbě aplikace bylo vytvořené grafického rozhranní pro tvorbu diagramů. Jednou ze silných stránek technologie Silverlight je výše zmíněná možnost vytvářet vlastní UserControly a dále s nimi pracovat jako s jakýmkoli jiným elementem.

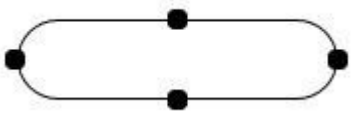
Jako první UserControly jsem vytvořil dva prototypy, které jsem nazval Element (prvek diagramu) a Relation (vztah mezi prvky diagramu). Silverlight UserControl sám o sobě neumožňuje uživatelům s ním pohybovat po obrazovce, ale každý element má naimplementován poměrně velké množství událostí spojených s akcemi uživatele. Díky těmto událostem bylo možné doimplementovat všechny potřebné metody, díky kterým spolu jednotlivé prvky fungují a přiřadit je k sobě.

Nejprve jsem začal s tvorbou Relationu. Každý Relation se skládá ze dvou malých obdeníků, určující konec a začátek vztahu a čáry mezi nimi. Konkrétní typy Relationů můžou mít různé další části, jako třeba popisek, nebo další čáry, tvořící šipku a podobně. Při kliknutí a tažení na některý z těchto objektů se Relation chová odlišně. Při tažení jednoho z obdélníků je tento obdélník tahán, zatímco druhý obdélník zůstává na svém místě a čára mezi nimi se dokresluje. Při tažení čáry se hýbe s celým Relationem.



Obr. 9: Relation

Základní stavbou Elementu je hlavní geometrický útvar s názvem Body. Na několika místech, podle potřeby, jsou na tomto útvaru vytvořeny malé obdélníky, které jsem nazval Control pointy. Elementy se svým tvarem a obsahem dost liší, mohou v nich být použity popisky, různé množství Control pointů a podobně. Pohyb Elementu je udělán klasicky (po kliknutí a táhnutí se pohybuje celý Element). Výše zmíněné Control pointy slouží k zachytávání Relationů na prvek.



Obr. 10: Element

Ve chvíli kdy je na některý Element, nebo Relation kliknuto a pohne se myš, provedou, vyvolají se příslušné události, které nastaví určité parametry a tím změní stav, nebo polohu prvku. Událost kliknutí rovněž nastaví parametr indikující, že je prvek tahán. Během této činnosti hlavní stránka kontroluje, zdali spolu některé prvky nezačaly komunikovat. Pokud je indikováno, že ano, jsou vyvolány potřebné metody, které různými způsoby pozmění další parametry prvků. Například pokud je detekováno, že Relation, který uživatel tahá, je nad některým z Elementů vyvolá se metoda, která nalezne nejbližší Control point, změní souřadnice jednoho z konce Relationu na souřadnice Control pointu a nastaví do parametrů Relationu

informaci, že je připnut k danému Elementu. Dokud není Relation stejným způsobem přesunut jinam, Element nad ním získává kontrolu a přesouvá se společně s ním.

Tímto principem jsou vytvořeny téměř všechny prvky a vztahy v diagramech. Pro komplikovanější prvky (jako třeba role u diagramu aktivit) jsem vytvořil takzvané hybridní prvky, kombinující vlastnosti obou výše zmíněných principů.

5.3.2 Propojení s databází

Jelikož technologie Silverlight neobsahuje .NET knihovny, potřebné k přímému připojení k databázi, bylo nutné využít jeden z alternativních způsobů, jak se dostat k potřebným datům.

Pro tuto aplikaci jsem zvolil metodu s využitím webové služby Silverlight-enabled WCF Service. Jelikož byla tato služba vytvořena v rámci ASP.NET části aplikace, bylo možné využít všechny .NET knihovny potřebné k přístupu k databázím (System.Data). V této službě jsem naimplementoval všechny potřebné metody. Po přidání Service Reference v samotném Silverlight projektu bylo možné výše zmíněné metody plně využívat.

5.3.3 Banka otázek

Banka otázek je velice důležitý pojem v oblasti vytváření testových aplikací. Testy se nevytváří celé najednou. Nejprve jsou vytvořeny samostatné otázky, které jsou uloženy, popřípadě zařazeny do kategorií, či jinak označeny a uloženy. V případě tohoto programu jsou jako kategorie myšleny typy diagramů plus klasická otázka. Uživatel poté přiřazuje jednotlivé otázky k testům podle potřeby. Díky tomu je možné použít jednu otázku ve více testech.

5.3.4 Princip kontroly

Studenti, kteří vyplní test a uloží jej, zároveň spustí algoritmus kontroly. Data potřebná ke kontrole má aplikace již uložena v sobě (originál při načítání testu a studentskou kopii při jejím uložení). Algoritmus projíždí kolekci prvků z diagramu, který vytvořil student a porovnává jednotlivé prvky s originálem. Pokud se neshodují, je odečten příslušný počet bodů. Počet bodů je dán učitelem při tvorbě každého diagramu. Při neshodě je rovněž změněna barva elementu v testu vytvořeným studentem, aby bylo při pozdější kontrole jasně viditelné, kde byla chyba. Po kontrole je celý test i s výsledkem uložen.

5.4 Prostředí aplikace v režimu „Učitel“

Režim učitele slouží profesorům, kteří vytváří a spravují testy v systému. Každý učitel má svůj vlastní účet, avšak může měnit diagramy a testy vytvořené jinými učiteli. Učitelé mají na výběr několik záložek:

5.4.1 Testy

- Tato záložka slouží ke správě celých testů
- Uživatel má možnost vytvořit nový test, nebo editovat některý z testů, které již byly vytvořeny v systému
- Do každého testu jsou pomocí tzv. banky otázek vloženy dříve vytvořené diagramy a k nim je přiřazená klasická otázka, která násobí body za správně doplněný diagram
- Ke každému testu může uživatel přiřazovat studenty
- Každý test je zabezpečen heslem, které studenti obdrží na místě konání testu

5.4.2 Výsledky

- Tato záložka slouží k prohlížení výsledků jednotlivých studentů.
- Všechny testy, které student absolvuje, jsou ukládány a učitelé si je zde mohou prohlédnout
- Testy studentů už není možné dále měnit

5.4.3 Diagramy aktivit, Sekvenční diagramy, Diagramy případů užití a Diagramy tříd

- Tyto záložky slouží k editaci a tvorbě nových diagramů (v závislosti na zvoleném typu diagramu), které jsou po uložení vloženy do banky otázek.
- Prvky, které mají studenti doplnit je třeba označit jako „Testovaný prvek“
- Učitelé musí u každého diagramu doplnit, kolik bodů se má za jakou chybu odečíst

5.4.4 Klasické otázky

- Tato záložka slouží k editaci a tvorbě nových otázek, které jsou po uložení vloženy do banky otázek.
- Otázky mají vždy několik možných odpovědí, správná je pouze jedna

- Tyto otázky můžou být umístěny za kterýkoliv diagram a tím zdvojnásobit bodový zisk z tohoto diagramu

5.5 Prostředí aplikace v režimu „Student“

Režim „Student“ slouží uživatelům, kteří mají být absolvovat některý z testů v systému. Tento režim má na výběr dvě záložky:

5.5.1 Aktuální test

- Po kliknutí na tuto záložku se uživateli zobrazí formulář pro zadání hesla k aktuálnímu testu
- Heslo by mělo být sděleno studentům na místě konání zkoušky
- Po zadání správného hesla je zobrazen celý test, který musí studenti vyplit

5.5.2 Absolvované testy

- V této záložce si studenti mohou prohlížet dříve absolvované testy
- Studenti vidí pouze bodové zisky z jednotlivých úkolů

5.6 Shrnutí

Poslední kapitola pojednává o problémech spojených s implementací a algoritmech, které tyto problémy řeší. Velkou překážkou byl poměrně malý zdroj informací týkající se technologie Silverlight, způsobený pravděpodobně poměrným mládím této technologie a velkými změnami mezi jednotlivými verzemi. Během této části mi byly neocenitelným pomocníkem různé fóra, na kterých byly řešeny různé problémy spojené s programováním v jazyce C# a XAML. Dalším mocným zdrojem informací byl web MSDN.

V druhé části této kapitoly jsem se zmínil o základním chování aplikace a tím i o možnostech jednotlivých druhů uživatelů.

6 Závěr

V této bakalářské práci jsme se zabývali vývojem aplikace sloužící k hodnocení studentů z oblasti softwarového inženýrství se zaměřením na jazyk UML.

V teoretické části bylo popsáno, co to vlastně softwarové inženýrství je a čím se zabývá předmět Úvod do softwarového inženýrství. Dále jsem popsal několik nejznámějších programů, používajících se ke kontrole studentů, zaměřené na společné základy a odlišnosti jednotlivých prvků. Z této kapitoly bylo vidět, že v tomto odvětví se používá několik stejných principů a odlišnosti mezi jednotlivými programy jsou spíše obsahového a kosmetického významu. Tyto poznatky mi pomohly při samotném návrhu aplikace.

Dále jsme se v teoretické části dozvěděli něco o základních technologiích použitých při implementaci a o důvodech, proč jsem je zvolil. Tato volba byla velmi ovlivněna mými předešlými zkušenostmi s rodinou technologií .NET, ze které jsem dále vybíral konkrétní technologii pro tento druh úlohy. Zvláštní pozornost jsem poté věnoval technologii Silverlight, kterou jsem si k účelu implementace vybral. Nakonec jsem také uvedl několik informací o jazyce UML, na který je kladen důraz při testování.

V praktické části jsem provedl analýzu požadavků a zmínil jsem problémy při implementaci a jejich řešení a s tím související přínos k velkému množství algoritmů moderního programovaní.

Nicméně hlavním výsledkem této práce je samotná aplikace, která podle mého názoru splňuje požadavky pro testování studentů v oblasti softwarového inženýrství a bude užitečným přínosem pro předmět zabývající se touto tématikou. Doufám, že bude brzy nasazena do provozu a že bude bezproblémově plnit svůj účel na Vysoké škole báňské – Technické univerzitě Ostrava.

Přínos této práce pro mě osobně byl v částečném zdokonalení znalostí jazyka UML a nejvíce ve velkém zdokonalení v programování na platformě .NET s využitím technologie Silverlight.

Literatura

1. *IS Edison*[online]. [citováno 26. Července 2011]. Dostupné z <http://as.wps.sso.vsb.cz/cz.vsb.edison.edu.study.prepare.web/SubjectVersion.faces?version=456-0517/01&studyPlanId=10718&locale=cs>

2. VONDRÁK, Ivo. *Úvod do softwarového inženýrství*[online]. c2002, [citováno 26. Července 2011]. Dostupné z <http://vondrak.cs.vsb.cz/download/Uvod_do_softwaroveho_inzenyrstvi.pdf>

3. *.NET Framework Overview*[online]. [citováno 28. Července 2011]. Dostupné z <http://www.microsoft.com/net/overview.aspx>

4. *Microsoft Web | Silverlight*[online]. [citováno 28.Července 2011]. Dostupné z <http://www.microsoft.com/cze/web/silverlight/>

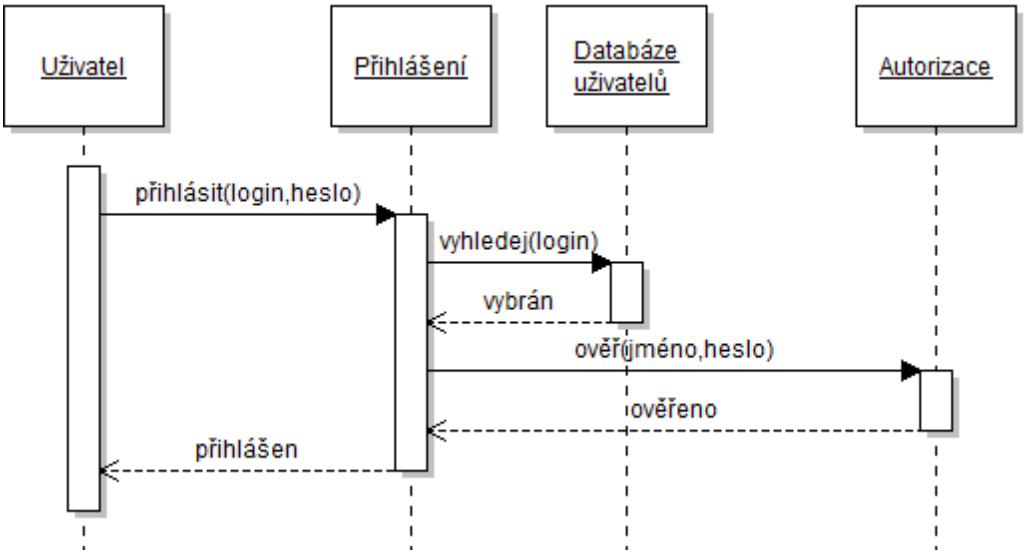
5. KOLDA, J. - *Microsoft Silverlight 2.0.* České Budějovice, 2009. 61 s. Příloha k bakalářské práci na Pedagogické fakultě Jihočeské university v Českých Budějovicích. Vedoucí práce PaedDr. Petr Pexa

6. PIK, Ivan - *Nástroje pro tvorbu layoutu v Silverlightu 2.0 a Silverlight toolkitu*[online]. [citováno 8.srpna 2011]. Dostupné z <http://zdrojak.root.cz/clanky/tvorba-layoutu-v-silverlightu-2-a-toolkitu/>

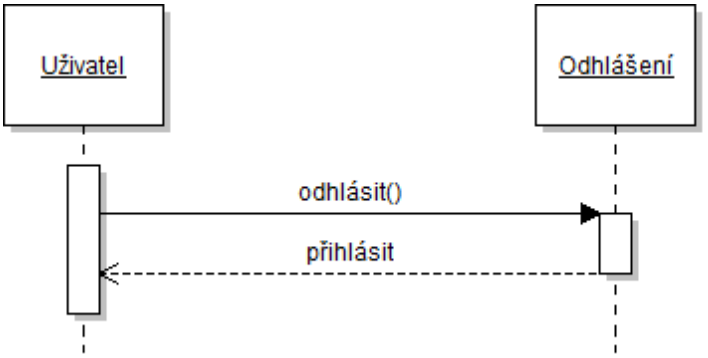
Seznam příloh

Příloha A: sekvenční diagramy

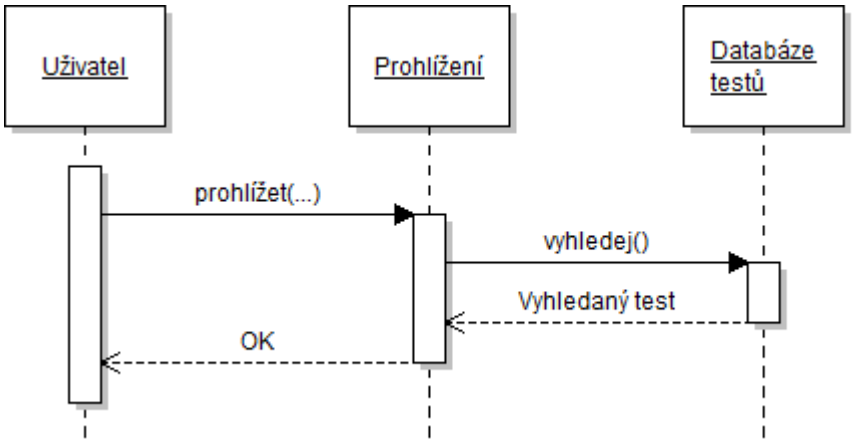
Přihlášení do systému:



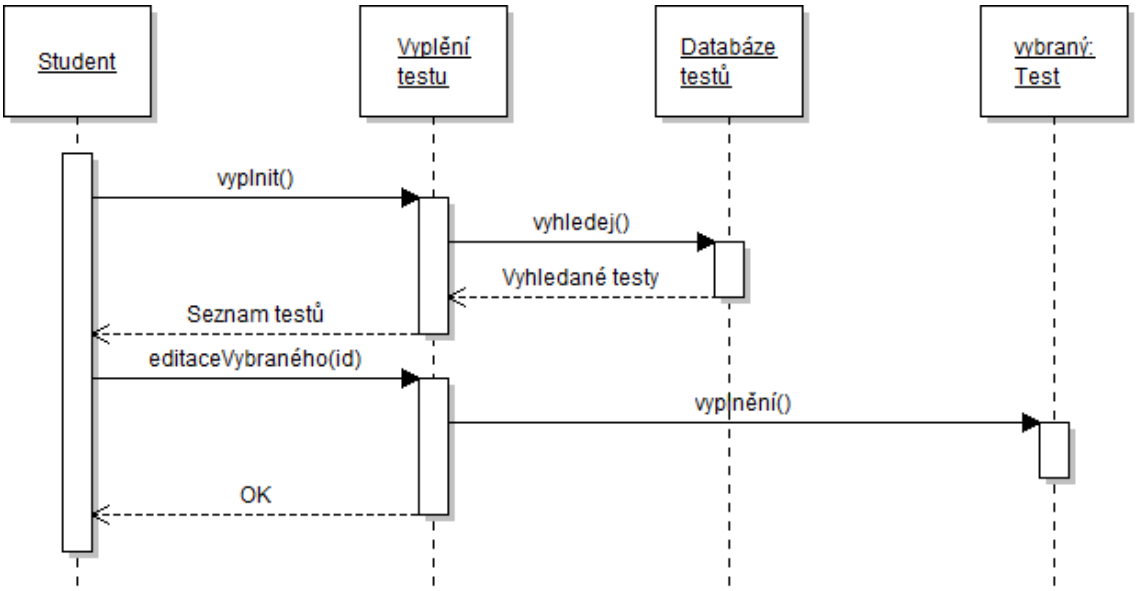
Odhlášení ze systému:



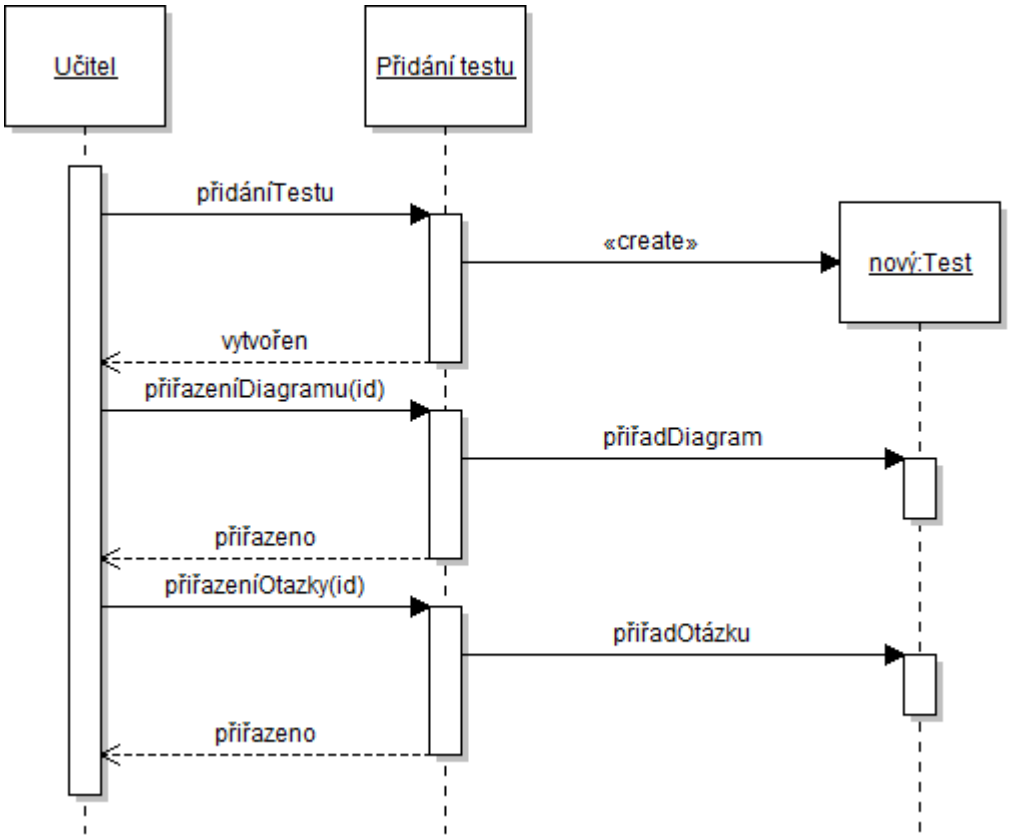
Prohlížení výsledků:



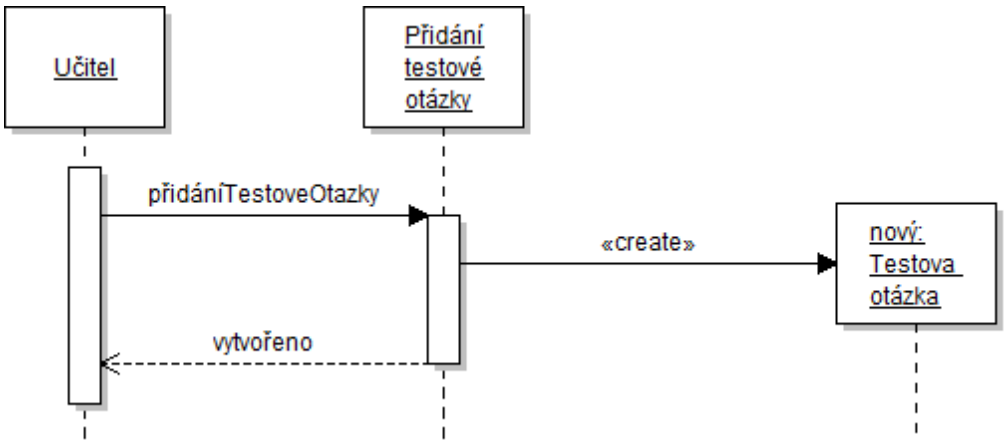
Vyplnění testu:



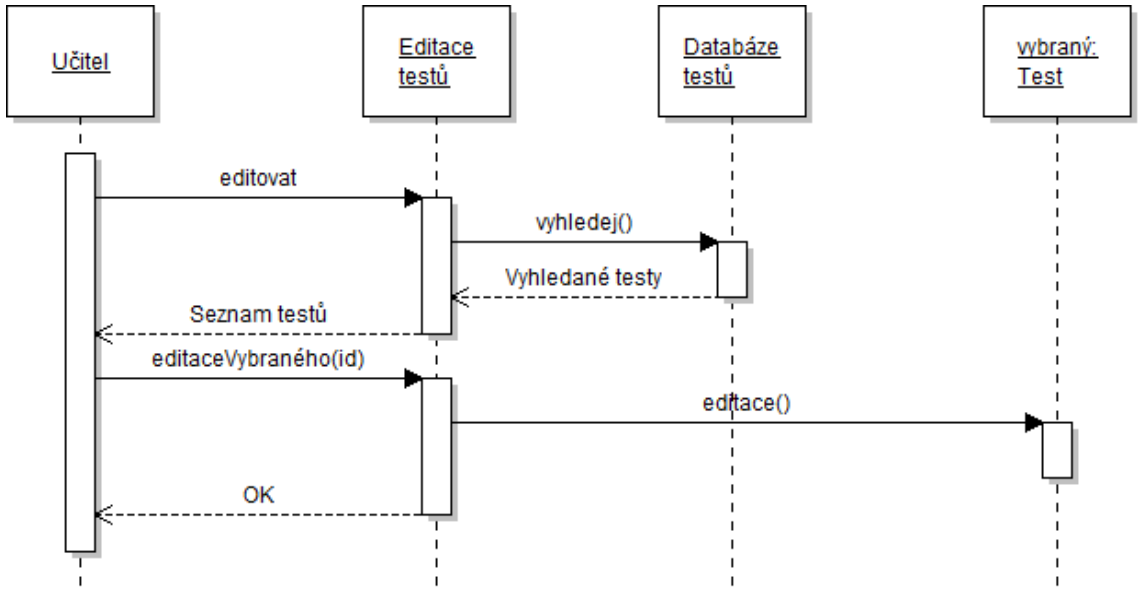
Vytvoření testu:



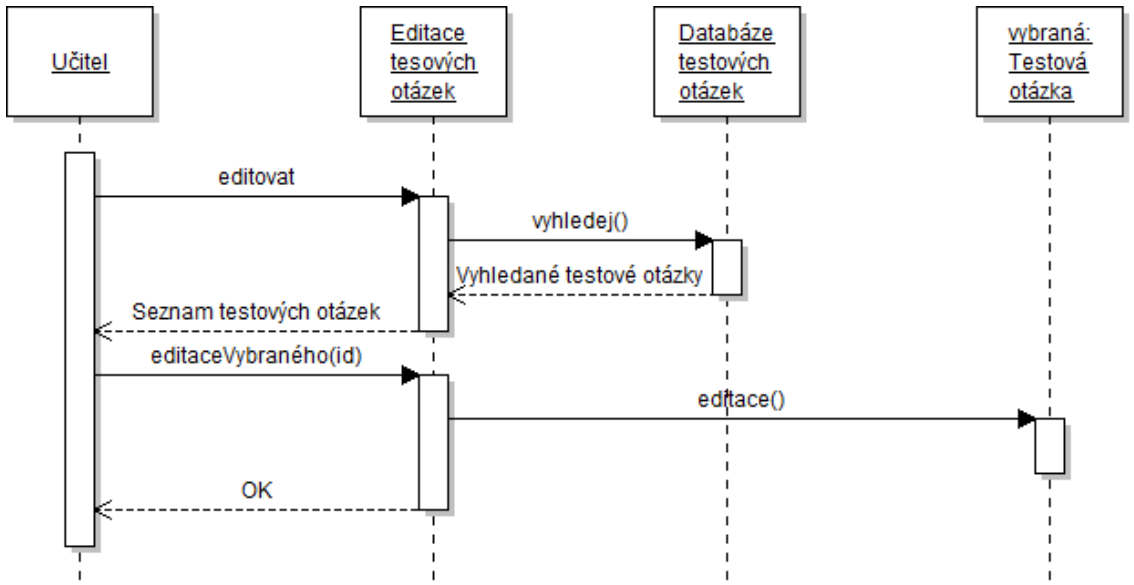
Vytvoření testové otázky:



Editace testu:



Editace testové otázky:



Přílohy na CD

1. Zdrojové soubory aplikace UML Tester
2. Uživatelská příručka k aplikaci UML Tester
3. Zdrojové kódy a návrh databáze
4. Elektronická verze bakalářské práce